Intelligent Input/Output LCD Module

(iIO-**LCM**)

User Guide

Version: 1.2

www.ibt.ca

eMail: info@ibt.ca

Toll Free: 1(866) 590-4288

iBT Technologies inc.

Tel. (514) 832-0808 Fax. (514) 832-0128

Table of Contents

Chapter 1 Introduction

The iIO (intelligent IO) Keypad LCD module is designed to provide a convenient management and control interface application ready hardware platforms and network appliances or headless controler system.

Our proprietary design offers an OS independent serial device for communication and control of computing systems (boards) where both LCD display and input keypad are required.

Through the serial interface, our solution offers a simple protocol which is defined so that applications can directly communicate with this module regardless of what Operating System it runs on.

WARNING!

THE LCD DRIVER IS MADE OF CMOS PROCESS AND CAN BE DAMAGED BY STATIC CHARGE VERY EASILY. MAKE SURE THE USER IS GROUNDED WHEN HANDLING THE LCD.

Chapter 2 Features

- Well-designed user interface for Network Appliance applications
- Driver free; OS independent
- 16x2 Alphanumeric characters display
- Five key pads can be customized for different applications
- Simple system installation and operation
- Function Key for “Hot Key” implementations

Chapter 3 Mechanical Specification

Module Size (mm):	101.6(W) x 26.0(H) x 28(D) (max.)
Display Format:	16 characters x 2 lines
Character Size:	2.96 x 5.56 mm

Chapter 4 General Specification

Display Resolution:	16 characters x 2 lines
Dimensional Outline (mm):	101.6(W) x 26.0(H) x 28(D) (max.)
Function Key:	Five operation keys (Up, Down, Enter ,ESC and Function)
Interface:	RS-232

Ambient Temperature	**Operating**		**Storage**	
	Max.	**Min.**	**Max.**	**Min.**
	0ºC	+50ºC	-20ºC	+70ºC

Chapter 5 Pin Assignment

There are only two connectors in this module (***Figure 5-1)***: power connector and serial port connector. ~~The power source into this module is 5 volt only.~~ 12 volts and 5 volts need to be present. There are only three pins used in the serial port interface (***Figure 5-2***).

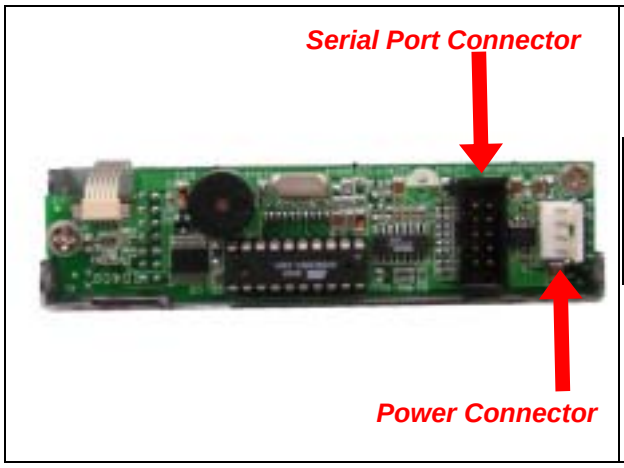


Fig. 5-1 *Power connector and serial port connector of iIO*

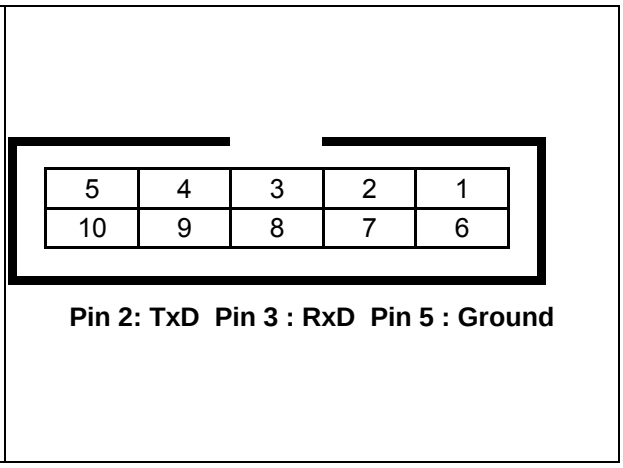


Fig. 5-2 *Pin assignment*

(1) Interface Pin Assignment

PIN NO.	PIN OUT	Description
1	NC	No connector
2	RXD	RS232 Data
3	TXD	Data RS232
4	NC	No connector
5	V_{SS}	Ground
6	NC	No connector
7	NC	No connector
8	NC	No connector
9	NC	No connector
10	NC	No connector

(2) Power

PIN NO.	PIN OUT	Description
1	NC	Power VCC (+12v)
2	GND	Power GND
3	GND	Power GND
4	+5V	Power VCC (+5V)

iIO installation steps :
• Connect the power connector to the power connector of this module.
• Connect the straight-through cable between Serial Port of this module and computer.

Chapter 6 iIO Function Command

iIO Command List Table			
Command	**Variable-length messages**	**Description**	**Response**
90H	Command byte	Write command to iIO as an internal operation.	Byte
92H	Data byte	Write 1 byte data to iIO (DDRAM / CGRAM)	Byte
93H	None	Read data from iIO (DDRAM / CGRAM)	Byte
94H	None	Enquire the response form iIO	Byte
95H	00H: Off ~ 0AH: fully light	Set up iIO backlight intensity	Byte
96H	00H: Buzzer off 01H: Buzzer on 02H: Enable key-stroke beep 03H: Disable key-stroke beep	00H: Turn buzzer off (default) 01H: Turn buzzer on 02H: Enable key-stroke beep (default) 03H: Disable key-stroke beep	Byte
97H	Easy String Display	Write multiple bytes data to iIO (DDRAM / CGRAM)	Byte

The following are examples for every command.
For clarity, the syntax is not exactly written in C language.
For detail program guide, please see the sample code below.
ttyS0 : COM1

90H:

Sample for clear the screen on iIO.

```
write(/dev/ttyS0,0x90,1);// send 90H command function.
write(/dev/ttyS0,0x01,1);// send iIO LCD function.
                         //Please see the iIO LCD Function Table.
checksum = 0x90 + 0x01;
checksum &= 0x7F; //We won't use bit 7.
write(/dev/ttyS0, checksum,1); // send checksum.
```

92H:

Sample for display character ' i ' on the iIO screen.

```
write(/dev/ttyS0,0x92,1);// send 92H command function.
messages[ ] = "i"; // Display the messages.
write(/dev/ttyS0, messages[0],1); // send messages[0].
checksum = 0x90 + messages[0];
checksum &= 0x7F; //We won't use bit 7.
write(/dev/ttyS0, checksum,1); // send checksum.
```

93H:

Sample for read the data from address 40H of iIO.

```
write(/dev/ttyS0,0x90,1);// send 90H command function.
write(/dev/ttyS0,0xC0,1);// Point to 40 address.( 0x80 + 0x40 = 0xC0 )
                         // Please see Note 1.
checksum = 0x90 + 0x40;
```

checksum &= 0x7F; //We won't use bit 7.
write(/dev/ttyS0, checksum,1); // send checksum.

//start to read the data from iIO.
write(/dev/ttyS0,0x93,1);// send 93H command function.
checksum = 0x93 & 0x7F; //We won't use bit 7.
write(/dev/ttyS0, checksum,1); // send checksum.
read(/dev/ttyS0,buffer,1);// read the data from iIO.

buffer[0]: The data of iIO is resided in buffer.

94H:

Sample for read the response from iIO.

write(/dev/ttyS0,0x94,1);// send 94H command function.
checksum = 0x94 & 0x7F; //We won't use bit 7.
write(/dev/ttyS0, checksum,1); // send checksum.
read(/dev/ttyS0,buffer,2);// Now, the response data from iIO would be resided in buffer.

buffer[0] : The response data from iIO is 1 Byte.
Please see the iIO Response Data List Table.
buffer[1] : The checksum data from iIO is 1 Byte and the format is (buffer[0] & 0x7F).
example :
buffer[0] = 0x90;// ESC key is pressed.
buffer[1] = buffer[0] & 0x7F;
buffer[1] = 0x10;// The checksum data.

95H:

Sample for set the brightness of iIO backlight.
Brightness : 00H (dark) ~
0AH (bright)

brightness = 0xA;// fully light
write(/dev/ttyS0,0x95,1);// send 95H command function.
write(/dev/ttyS0,brightness,1);// send the brightness we want.
Checksum = 0x95 + brightness;
checksum &= 0x7F; //We won't use bit 7.
write(/dev/ttyS0, checksum,1); // send checksum.

96H:

Sample for turn on the buzzer of iIO.
Buzzer : 00H (off) , 01H (on)

buzzer = 0x01// Turn on the buzzer.
write(/dev/ttyS0,0x96,1);// send 96H command function.
write(/dev/ttyS0,buzzer,1);// set the buzzer we want.
Checksum = 0x96 + buzzer;
checksum &= 0x7F; //We won't use bit 7.
write(/dev/ttyS0, checksum,1); // send checksum.

97H:

This command can be used to display string and cursor easily at any location.
We strongly recommend this command if you want to display string on **iIO** screen.
The usage is as following:

Packet length in bytes + Cursor & Blinking mode + Display initial location + Variable-length string

Packet length in bytes (05H ~ 8FH) +
Cursor & Blinking mode (
00H : Cursor off & Blinking off ,

01H : Cursor off & Blinking on ,
02H : Cursor on & Blinking off ,
03H : Cursor on & Blinking on)+

Display initial location (
Row 1 : 00H ~ 0FH ,
Row 2 : 40H ~ 4FH)+

Variable-length string (None ~ 16 characters)

Example :

```
cursor_blinking = 0x3; // cursor on & blinking on.
init_addr = 0x40; // Display initial location on row 2.
messages[  ] = “ibt”; // Display the messages.
messages_length = strlen(messages); // The messages total characters length.
packet_length = messages _length + 5; // see Note 2.
checksum = 0x97+ cursor_blinking + init_addr + packet_length + messages[0] + messages[1]
           messages[2] + messages[3] + messages[4] ;
checksum &= 0x7F; // Now, we start to send every character to iIO.

write(/dev/ttyS0,0x97,1); // send 97H command function.
write(/dev/ttyS0,packet_length,1); // send  total packet  length.
write(/dev/ttyS0, cursor_blinking,1); // send cursor & blinking mode.
write(/dev/ttyS0, init_addr,1); // send initial address.
write(/dev/ttyS0, messages[0],1); // send messages[0].
write(/dev/ttyS0, messages[1],1); // send messages[1].
write(/dev/ttyS0, messages[2],1); // send messages[2].
write(/dev/ttyS0, messages[3],1); // send messages[3].
write(/dev/ttyS0, messages[4],1); // send messages[4].
write(/dev/ttyS0, checksum,1); // send checksum.
//Done.
```

Note 1:

For 90 / 92 / 93H command, we use the LCD built-in command.
The address (see the iIO LCD Address List Table) must be added 1 on bit 7 after fill in the address you want.
Example :
At the first column in the second row, the address is 40H.
You have to add 80H (bit 7 is 1).
So, the address is C0H. (40H + 80H)
Please see the “ set the DDRAM address “on the iIO LCD function table.

Note 2:

97H command + Packet length +Cursor & Blinking mode + Display initial location + Checksum
1Byte +1Byte +1Byte +1Byte +1Byte = 5Bytes

Note 3:

- Turn on/off the backlight through “gradual bright / dark” procedure.
- The activity of Fn key will be automatic.
  The functionality of Fn key will be canceled after 5 seconds without any key pressed.
- PC must wait response from iIO to avoid messages overlapped.
- Turn off iIO backlight after a certain time without any operation to extend the life time of backlight.
- One PC enquiry must come along with one iIO response.
- Values from 90H to 9FH in both Command byte and Data byte are all reserved for MHDs(Message-HeaD byte).
  They aren’t permitted to be used in both Command byte and Data byte.
- Clear the bit 7 of CKS byte after CKS accumulation.
- Response to PC – Timeout : Communication data lost

Non-AcKnowledge(0x9A) : Checksum erred
iIO to PC checksum error : Command unstable.

iIO Response Data List Table		
Response Byte	**iIO Keypad**	**Description**
90H	ESC	ESC key on iIO keypad
None	Fn	Fn key on iIO keypad
92H	Up arrow	Up arrow key on iIO keypad
93H	Down arrow	Down arrow key on iIO keypad
94H	Enter	Enter key on iIO keypad
95H	F1 (Fn →Up arrow)	Press Fn key once, then touch the Up arrow key.
96H	F1 (Fn → Down arrow)	Press Fn key once, then touch the Down arrow key.
97H	F1 (Fn → Enter)	Press Fn key once, then touch the Enter key.
98H	None	No any key is pressed.
9AH	None	(Non-AcKnowledge) Checksum error.

iIO LCD Address List Table																
Character	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Location (Address)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

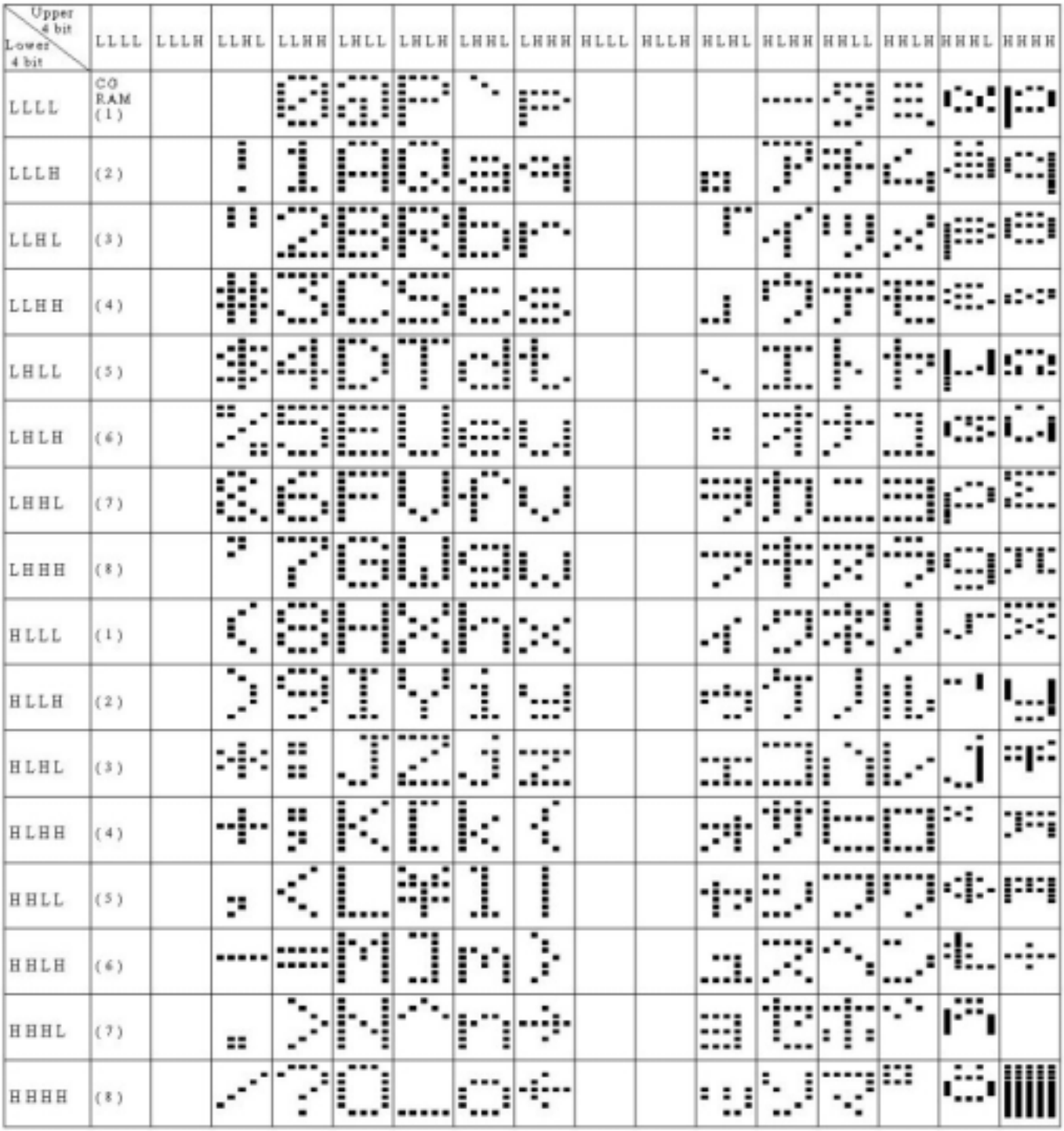
The input must be a standard RS-232 or inverted TTL signal. The RS-232 setting should be:

- *Baud rate: 57600 bps*
- *Parity: None or even parity*
- *Data bits: 8*
- *Stop bit: 1*

iIO LCD Function List Table

Instruction	Command write / Data read										Description	Execution time (fosc=270Khz)
			DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display			0	0	0	0	0	0	0	1	Write "00H" to DDRAM and set DDRAM address to "00H" from AC	1.53ms
Return Home			0	0	0	0	0	0	1	—	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.53ms
Entry Mode Set			0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display.	39 μs
Display ON/OFF Control			0	0	0	0	1	D	C	B	Set display (D), cursor (C), and blinking of cursor (B) on/off control bit.	39 μs
Cursor or Display Shift			0	0	0	1	S/C	R/L	—	—	Set cursor moving and display shift control bit, and the direction, without changing of DDRAM data.	39 μs
Function Set			0	0	1	DL	N	F	—	—	Set interface data length (DL:8-bit/4-bit), numbers of display line (N:2-line/1-line)and, display font type (F:5×11 dots/5×8 dots)	39 μs
Set CGRAM Address			0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter.	39 μs
Set DDRAM Address			1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter.	39 μs
Read Busy Flag and Address			BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 μs
Write Data to RAM			D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43 μs
Read Data from RAM			D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43 μs

Chapter 7 Character Generator ROM (CGROM)



main.c

```
/*
 * iBT intelligent IO LCD Module Sample:
 *
 * Copyright (C) 2004-2005 iBT Inc.
 * Copyright (C) 2004-2005 Chris Chiu
 *
 *   This program is free software; you can redistribute it and/or modify
 *   it under the terms of the GNU General Public License as published by
 *   the Free Software Foundation; either version 2 of the License, or
 *   (at your option) any later version.
 *
 *   This program is distributed in the hope that it will be useful,
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *   GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *   along with this program; if not, write to the Free Software
 *   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.
 */

//-------------------------------------------------------------------------
// iIO Communication Protocol
//-------------------------------------------------------------------------
/*

PC(Master) - to - iIO(Slave) request:
   (Message Head byte) + (variable_length Message byte) + (Checksum byte)

iIO(Slave) - to - PC(Master) response:
   (Response byte) + (Checksum byte)

(checksum byte) = (message head byte) + (varibale_length message byte)

Note:See protocol.h
*/

#include <time.h>
#include <stdlib.h>
#include <string.h>
#include <termios.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <ctype.h>
#include <stdio.h>

#include "protocol.h"
#include "function.h"

extern int serial_init(int port, int baud_rate);
extern void uninit_serial( void );
extern void cmd_sendto_lcm( unsigned int function );
extern void msg_sendto_lcm( unsigned int addr , char *msg );
```

```
extern void backlight( unsigned int light );
extern void buzzer( unsigned int flag );
extern void polling( unsigned char *buf1 , unsigned char *buf2 );
extern void read_data( unsigned int addr , unsigned char *buf1 , unsigned char *buf2 );
extern void smart_display( unsigned int cursor_mode , unsigned int addr , unsigned char *msg );

#define VERSION "1.0"

//com port. 0=COM1/ttyS0 1=COM2/ttyS1
#define COMPORT  0
//baud rate
#define BAUD 57600

int fd;

int main(int argc, char* argv[])
{

  unsigned int addr;
  unsigned char buf[2];
  unsigned char c[] = "i";

  /* Add or Change Show Message here */
  unsigned char msg_title[] = "iBT Tech Inc.";
  unsigned char msg_up[] = "(Received UP)   ";
  unsigned char msg_down[] = "(Received DOWN) ";
  unsigned char msg_enter[] = "(Received ENTER)";
  unsigned char msg_esc[] = "(Received ESC)  ";
  unsigned char msg_fn_up[] = "FN rebooting....";
  unsigned char msg_fn_down[] = "FN shutdown.....";
  unsigned char msg_fn_enter[] = "Display MAINmenu";

  fprintf(stdout,"iBT Inc. iIO communications sample " VERSION"\n");

  if ( serial_init(COMPORT, BAUD) )  /* Initialize RS-232 environment */
     printf("Serial port ttyS%d error !\n",COMPORT);


  printf("90H test: Clear display ...\n");
   cmd_sendto_lcm(LCD_CLEAR);
   sleep(2);

  printf("92H test: Send 'i' character to iIO ....\n");
   addr=0xCF;
   msg_sendto_lcm(addr,&c);
   sleep(2);

  printf("95H test: iIO backlight testing ....\n");
   backlight(0x0);//Turn off the backlight.
   sleep(2);
   backlight(0xA);//Turn on the backlight.

  printf("96H test: iIO buzzer testing ....\n");
   buzzer(0x01);//Turn on the buzzer.
   sleep(2);
   buzzer(0x00);//Turn off the buzzer.

  printf("97H test: Send 'IBT Inc.' string to IO ....\n");
   cmd_sendto_lcm(LCD_CLEAR);
   smart_display(0x02,0x80,msg_title);
```

```
  for(addr=0x80;addr<=0x8f;addr++){

        read_data(addr,&(buf[0]),&(buf[1]));
        printf("Reading from iIO addr=0x%x char=%c\n",addr,buf[0]);
        usleep(500000);
  }

 printf("94H test: please press any key from iIO keypad ...\n");

  for(;;)  {

  polling( &(buf[0]) , &(buf[1]) );
        usleep(50000);

        switch( buf[0] & buf[1] ) {       /* Switch the Read command */

        case 0x10 :    /* ESC Botton was received */
                       smart_display(0x00,0x40,msg_esc);
                break;

        case 0x12 :    /** UP Botton was received */
                       smart_display(0x01,0x40,msg_up);
                break;

        case 0x13 :    /** DOWN Botton was received */
                       smart_display(0x02,0x40,msg_down);
                break;

        case 0x14 :    /** ENTER Botton was received */
                       smart_display(0x03,0x40,msg_enter);
                break;

        case 0x15 :    /** Fn_UP Botton was received */
                       //You could call your function here.
                       smart_display(0x03,0x40,msg_fn_up);
                break;

        case 0x16 :    /** Fn+DOWN Botton was received */
                       //You could call your function here.
                       smart_display(0x03,0x40,msg_fn_down);
                break;

        case 0x17 :    /** Fn+ENTER Botton was received */
                       //You could call your function here.
                       smart_display(0x03,0x40,msg_fn_enter);
                break;

        default   :
                sync();
                break;
        }

 }

  uninit_serial();

  exit(0);

}
```

protocol.h

```
/* intelligent IO command function */
#define IIO_COMMAND 0x90 //Command head byte
#define IIO_WRITE_DATA      0x92 //Write data to CGRAM.
#define IIO_READ_DATA       0x93 //Read data from CGRAM.
#define IIO_RESPONSE        0x94 //Enquire LCM response.
#define IIO_BACKLIGHT       0x95 //Setup LCM backlight intensity.
#define IIO_BUZZER    0x96 //Turn on/off buzzer & Enable/Disable key-stroke beep.
#define IIO_SMART_DISPLAY 0x97 //Command head byte

/* LCD function */
#define LCD_CLEAR    0x01 // Clear screen on LCM.
#define LCD_BLANK    0x08 // The LCM screen blank.
#define LCD_HIDE     0x0C // Hide the cursor and display blanked characters.
#define LCD_TURNON  0x0D // Blanking block cursor.
#define LCD_UNDERLINE       0x0E // Show underline cursor.
#define LCD_MV_LEFT 0x10 // Move cursor 1 character left.
#define LCD_MV_RIGHT        0x14 // Move cursor 1 character right.
#define LCD_SCRL_LEFT       0x18 // Scroll cursor 1 character left.
#define LCD_SCRL_RIGHT      0x1C // Scroll cursor 1 character right.

/* Receiving the Correspinding Status. */
#define RECV_ESC      0x90 //Receive the ESC button.
#define RECV_UP             0x92 //Receive the Up button.
#define RECV_DOWN   0x93 //Receive the Down button.
#define RECV_ENTER  0x94 //Receive the Enter button.
#define RECV_FN_UP  0x95 //Receive the Function+Up.
#define RECV_FN_DOWN        0x96 //Receive the Function+Down.
#define RECV_FN_ENTER       0x97 //Receive the Function+Enter.
```

function.h

```
extern int serial_init(int port, int baud_rate);
extern void uninit_serial(void);
extern int recv_response(void);
extern void cmd_sendto_lcm( unsigned int function );
extern void msg_sendto_lcm( unsigned int addr , char *msg );
extern void backlight( unsigned int light );
extern void buzzer( unsigned int flag );
extern void polling( unsigned char *buf1 , unsigned char *buf2 );
extern void read_data( unsigned int addr , unsigned char *buf1 , unsigned char *buf2 );
extern void smart_display( unsigned int cursor_mode , unsigned int addr , unsigned char *msg );
```

function.c

```
#include       <stdlib.h>
#include       <string.h>
#include       <termios.h>
#include       <unistd.h>
#include       <fcntl.h>
#include       <errno.h>
#include       <ctype.h>
#include      <stdio.h>
#include      <sys/time.h>
#include      "function.h"
#include      "protocol.h"

#define FALSE   0
#define TRUE    1


// serial device name
const char *serial_port_names[]=
      {"ttyS0", "ttyS1", "ttyS2", "ttyS3", "ttyS4",
      "ttyS5", "ttyS6", "ttyS7", "ttyS8", "ttyS9"};

// com port handle
int fd;
struct timeval old_tv, new_tv;

//-------------------------------------------------------------------------
int serial_init(int port, int baud_rate)
{
      char devname[512];
      int brate;
      struct termios term;

      //make device name string
      sprintf(devname, "/dev/%s", serial_port_names[port]);

      //open device
      fd = open(devname, 2);
      if (fd <= 0)
      {
            //error opening port
            printf("unable to open device\n");
            return(1);
      }

      //get baud rate
      switch (baud_rate)
      {
            case 1200:
                  brate=B1200;
                  break;
            case 1800:
                  brate=B1800;
                  break;
            case 2400:
                  brate=B2400;
                  break;
            case 4800:
```

```
                brate=B4800;
                break;
        case 9600:
                brate=B9600;
                break;
        case 19200:
                brate=B19200;
                break;
        case 38400:
                brate=B38400;
                break;
        case 57600:
                brate=B57600;
                break;
        case 115200:
                brate=B115200;
                break;
        default:
                printf("invalid baud rate: %d\n", baud_rate);
                return(2);
}

//get device struct
if (tcgetattr(fd, &term) != 0)
{
      printf("tcgetattr failed\n");
      return(3);
}

//input modes
term.c_iflag &= ~(IGNBRK|BRKINT|PARMRK|INPCK|ISTRIP|INLCR|IGNCR|ICRNL
          |IUCLC|IXON|IXANY|IXOFF|IMAXBEL);
term.c_iflag |= IGNPAR;
//term.c_iflag |= (INPCK|PARMRK);// For even parity mode.

//output modes
term.c_oflag &= ~(OPOST|OLCUC|ONLCR|OCRNL|ONOCR|ONLRET|OFILL
          |OFDEL|NLDLY|CRDLY|TABDLY|BSDLY|VTDLY|FFDLY);
term.c_oflag |= NL0|CR0|TAB0|BS0|VT0|FF0;

//control modes
term.c_cflag &= ~(CSIZE|PARENB|CRTSCTS|PARODD|HUPCL);
term.c_cflag |= CREAD|CS8|CSTOPB|CLOCAL;//No parity
//term.c_cflag |= CREAD|CS8|CSTOPB|PARENB|CLOCAL;//Even parity
//term.c_cflag |= CREAD|CS8|CSTOPB|PARODD|CLOCAL;//Odd parity

//local modes
term.c_lflag &= ~(ISIG|ICANON|IEXTEN|ECHO|FLUSHO|PENDIN);
term.c_lflag |= NOFLSH;

//set baud rate
cfsetospeed(&term, brate);
cfsetispeed(&term, brate);

tcflush(fd,0);
tcflush(fd,1);
tcflush(fd,2);

//set new device settings
if (tcsetattr(fd, TCSANOW, &term)  != 0)
```

```
	{
		printf("tcsetattr failed\n");
		return(4);
	}

	return 0;
}

void uninit_serial(void)
{
	close(fd);
	fd=(int)NULL;
}

/* Check the response from iIO. */
int recv_response( void )
{

	unsigned char buf[2];
	unsigned int i;

	read(fd,buf,2);
	gettimeofday(&new_tv,NULL);

	i = buf[0] & buf[1];

	if ( (new_tv.tv_sec - old_tv.tv_sec) > 2 ){
		gettimeofday(&old_tv,NULL);
		printf("iIO not received any data from PC !\n");
		return 1;
	}else if( i == 0x1a ){
		gettimeofday(&old_tv,NULL);
		printf("iIO checksum error !\n");
		return 2;
	}else if( (buf[0] & 0x7f) != buf[1] ){
		gettimeofday(&old_tv,NULL);
		printf("PC checksum error !\n");
		return 3;
	}else{
		return 0;
	}


}

/* Send COMMAND to iIO. */
void cmd_sendto_lcm( unsigned int function )
{
	unsigned int cksum;
	unsigned int p=0;

	p = IIO_COMMAND;
	cksum =  IIO_COMMAND + function;
	cksum &= 0x7f;// Force the bit 7 will be '0'.
	write(fd,&p,1);// Send IIO_COMMAND to iIO.
	write(fd,&function,1);// Send IIO_COMMAND FUNCTION to iIO.
	write(fd,&cksum,1);// Send checksum to iIO.

	gettimeofday(&old_tv,NULL);
```

```
        while ( recv_response() ) {

                write(fd,&p,1);//Send IIO_COMMAND to iIO.
                write(fd,&function,1);//Send IIO_COMMAND FUNCTION to iIO.
                write(fd,&cksum,1);// Send checksum to iIO.
        }

}

/* Send the 1 character to iIO. */
void msg_sendto_lcm( unsigned int addr , char *msg )
{
        unsigned int cksum,i;
        unsigned int p,p1;

        /* addr : Set cursor address to 0x??.
                  We would like to display char on iIO.
        */

        p = IIO_COMMAND;
        cksum =  IIO_COMMAND + addr;
        cksum &= 0x7f;// Force the bit 7 will be '0'.

        write(fd,&p,1);// Send IIO_COMMAND to iIO.
        write(fd,&addr,1);// Send CURSOR ADDRESS FUNCTION to iIO.
        write(fd,&cksum,1);// Send checksum to iIO.
        gettimeofday(&old_tv,NULL);

        while ( recv_response() ) {

                write(fd,&p,1);// Send IIO_COMMAND to iIO.
                write(fd,&addr,1);// Send CURSOR ADDRESS FUNCTION to iIO.
                write(fd,&cksum,1);// Send checksum to iIO.
        }

        // Start to send characters.
        p1 = IIO_WRITE_DATA;

        for ( i=0 ; i< strlen(msg) ; i++ ){

                cksum =  IIO_WRITE_DATA + msg[i];
                cksum &= 0x7f; // Force the bit 7 will be '0'.
                write(fd,&p1,1); // Send WRITE_DATA COMMAND to iIO.
                write(fd,&(msg[i]),1); // Send char to iIO.
                write(fd,&cksum,1); //Send checksum to iIO.

                gettimeofday(&old_tv,NULL);

                while ( recv_response() ) {

                   write(fd,&p1,1); // Send WRITE_DATA COMMAND to iIO.
                   write(fd,&(msg[i]),1); // Send chararacter to iIO.
                   write(fd,&cksum,1); //Send checksum to iIO.
                }

        }

}

/* The easy way to display strings. */
```

```
void smart_display( unsigned int cursor_mode , unsigned int addr , unsigned char *msg )
{
	unsigned int cksum,len,i,j;
	unsigned int tmp=0;
	unsigned int p,p1;

	/* addr : Set cursor address to 0x??.
		We would like to display char on iIO.
	*/

	p = IIO_SMART_DISPLAY;
	p1 = cursor_mode;
	len = strlen(msg) + 5;

	for ( i=0 ; i< strlen(msg) ; i++ )
		tmp += msg[i];

	cksum =  IIO_SMART_DISPLAY + cursor_mode + addr + len + tmp;
	cksum &= 0x7f;// Force the bit 7 will be '0'.

	write(fd,&p,1);// Send SMART_DISPLAY to iIO.
	write(fd,&len,1);// Send packet length to iIO.
	write(fd,&p1,1);// Send cursor & blinking mode to iIO.
	write(fd,&addr,1);// Send CURSOR ADDRESS FUNCTION to iIO.

	for ( j=0 ; j< strlen(msg) ; j++ ){
		write(fd,&(msg[j]),1); // Send characters to iIO.
	}

	write(fd,&cksum,1);// Send checksum to iIO.

	gettimeofday(&old_tv,NULL);

	while ( recv_response() ) {

		write(fd,&p,1);// Send IIO_SMART_DISPLAY to iIO.
		write(fd,&len,1);// Send packet length to iIO.
		write(fd,&p1,1);// Send cursor & blinking mode to iIO.
		write(fd,&addr,1);// Send CURSOR ADDRESS FUNCTION to iIO.

		for ( j=0 ; j< strlen(msg) ; j++ )
			write(fd,&(msg[j]),1); // Send string to iIO.

		write(fd,&cksum,1);// Send checksum to iIO.

	}


}

/* Set the brightness of iIO backlight. */
void backlight( unsigned int light )
{
	unsigned int cksum;
	unsigned int p=0;

	p = IIO_BACKLIGHT;

	cksum =  IIO_BACKLIGHT + light;
	cksum &= 0x7f;// Force the bit 7 will be '0'.
```

```
        write(fd,&p,1);// Send BACKLIGHT command to iIO.
        write(fd,&light,1);// Send intensity to iIO.
        write(fd,&cksum,1);// Send checksum to iIO.
        gettimeofday(&old_tv,NULL);

        while ( recv_response() ) {

                write(fd,&p,1);// Send BACKLIGHT command to iIO.
                write(fd,&light,1);// Send intensity to iIO.
                write(fd,&cksum,1);// Send checksum to iIO.
        }
}

/* Turn ON/OFF the buzzer of iIO. */
void buzzer( unsigned int flag )
{
        unsigned int cksum;
        unsigned int p=0;

        p = IIO_BUZZER;

        cksum =  IIO_BUZZER + flag;
        cksum &= 0x7f;// Force the bit 7 will be '0'.
        write(fd,&p,1);// Send BUZZER command to iIO.
        write(fd,&flag,1);// Send (turn on/off buzzer) command to iIO.
                        // flag : 0x01 --- turn on buzzer
                        // flag : 0x00 --- turn off buzzer
        write(fd,&cksum,1);// Send checksum to iIO.

        gettimeofday(&old_tv,NULL);

        while ( recv_response() ) {

          write(fd,&p,1);// Send BUZZER command to iIO.
          write(fd,&flag,1);// Send (turn on/off buzzer) command to iIO.
                        // flag : 0x01 --- turn on buzzer
                        // flag : 0x00 --- turn off buzzer
          write(fd,&cksum,1);// Send checksum to iIO.
        }

}

/* Inquire the response data from iIO. */
void polling( unsigned char *buf1 , unsigned char *buf2 )
{
        unsigned int cksum;
        unsigned int p=0;
        unsigned char buf[2];

        p = IIO_RESPONSE;
        cksum = IIO_RESPONSE & 0x7f;// Force the bit 7 will be '0'.
        write(fd,&p,1);// Send the RESPONSE command to iIO.
        write(fd,&cksum,1);// Send checksum to iIO.

        read(fd,buf,2);

        *buf1 = buf[0];
        *buf2 = buf[1];

}
```

```
/* Read the data of iIO */
void read_data( unsigned int addr , unsigned char *buf1 , unsigned char *buf2 )
{
	unsigned int cksum;
	unsigned int p,p1;
	unsigned char buf[2];

	/* addr : Set cursor address to 0x??.
		We would like to get char from iIO.
	*/

	p = IIO_COMMAND;
	cksum =  IIO_COMMAND + addr;
	cksum &= 0x7f;// Force the bit 7 will be '0'.

	write(fd,&p,1);// Send IIO_COMMAND to iIO.
	write(fd,&addr,1);// Send CURSOR ADDRESS FUNCTION to iIO.
	write(fd,&cksum,1);// Send checksum to iIO.

	gettimeofday(&old_tv,NULL);

	while ( recv_response() ) {

		write(fd,&p,1);// Send IIO_COMMAND to iIO.
		write(fd,&addr,1);// Send CURSOR ADDRESS FUNCTION to iIO.
		write(fd,&cksum,1);// Send checksum to iIO.
	}

	p1 = IIO_READ_DATA;
	cksum = IIO_READ_DATA & 0x7f;// Force the bit 7 will be '0'.
	write(fd,&p1,1);// Send READ_DATA command to iIO.
	write(fd,&cksum,1);// Send checksum to iIO.

	read(fd,buf,2);

	*buf1 = buf[0];
	*buf2 = buf[1];

}
```